IN THE SPECIFICATION

Amendments to the Specification:

Please amend the paragraph beginning on page 25, line 11, as follows:

In operation 1014 ~~14~~, the top application's control module 306 starts other modules, J2EE Servers, and child applications. Starting of child applications leads to recursion, which results in starting all J2EE applications. Then, in operation 1015, the top user application's control module 306 reports completion of the start children request to the executive 210. In response, the executive 210 reports completion to the root J2EE server 1052, in operation 1016, and, in operation 1017, the root J2EE server 1052 reports completion to the bootstrap program 216 ~~the root system application's control module reports completion to the root J2EE server~~.

Please amend the paragraph beginning on page 38, line 8, as follows:

In operation 2104, the new Root J2EE Server 1052b loads its class files from the new J2EE Server Archive in the repository 606. The new Root J2EE Server 1052b then reports completion to the executive 210a, in operation 2105. Next, in operation 2106, the executive 210a ~~210~~ requests new Root J2EE Server 1052b to take over the executive module 210b ~~210a~~.

Please amend the paragraph beginning on page 40, line 4, as follows:

In operation 2204 ~~2104~~, the new Root J2EE Server 1052b loads the executive's RSM physical schema class files, and in operation 2205 ~~2105~~, the new Root J2EE Server 1052b upgrades the RSM state to the new physical schema. If necessary, the new Root J2EE Server

1052b requests the state server 614 to upgrade the executive's SSPartitions to the new schema, in operation 2206.

Please amend the paragraph beginning on page 40, line 9, as follows:

Then, in operation 2207, the new Root J2EE Server 1052b creates a new instance 210c of executive 210b, and loads the new executive's 210c upgrade class files from the repository 606, in operation 2208. The new Root J2EE Server 1052b then, in operation 2209 ~~2109~~, disables requests to the second executive 210b, which is the older version of the new executive 210c.

Please amend the paragraph beginning on page 24, line 11, as follows:

In operation 1004, the bootstrap program 216 starts a root J2EE Server 1052 process using an extension of the J2EE Server start sequence 900. In addition, in operation 1005, the bootstrap program 216 uses an extension of the J2EE Server start sequence 900 to start a root J2EE Server Backup 1054 process. Then, in operation 1006, the bootstrap program 216 requests root J2EE Server 1052 to start the root system application. The Root J2EE Server 1052 then starts the root system application by allocating the control module, which is the module implementing the executive 210, in operation 1007 ~~1008~~. After starting the root system application, the Root J2ee Server 1052 loads the executive's class files from the repository 606, in operation 1008 ~~1009~~, and, in operation 1009 ~~1010~~, the Root J2EE Server 1052 loads the executive's recoverable state from the state server 614.

Please amend the paragraph beginning on page 24, line 22, as follows:

In operation 1010 ~~1011~~, the Root J2EE Server 1052 requests the executive 210 to start the root system application. It should be noted that the executive 210 functions as the

control module for the root system application. In response, the executive 210 requests itself to start the root system application's other modules, J2EE servers, and child system applications using the module start sequence 800, J2EE Server start sequence 900, and application start sequence 700, in operation 1011. In this capacity, the executive 210 functions as the parent control module in these start sequences.

Please amend the paragraph beginning on page 34, line 9, as follows:

In operation 1712, the executive 210 requests the new control module 1752 to continue the upgrade. In response, the new control module 1752 upgrades all child J2EE Servers, in operation 1713. The new control module 1752 also upgrades all application service modules, in operation 1714, and upgrades all child applications by recursively using the application upgrade sequence 1700, in operation 1715 ~~1714~~.

Please amend the paragraph beginning on page 19, line 18, as follows:

Figure 7 is a sequence diagram showing an application start sequence 700, in accordance with an embodiment of the present invention. The application start sequence 700 illustrates how a parent application's control module 306 initiates the start of a child application by creating the child application's control module 750. Broadly speaking, the parent applications control module 306 allocates the child application's control module 750 to a specified J2EE server 502. The child application's control module 750 then coordinates the remainder of the child application's start up.

Please amend the paragraph beginning on page 29, line 10, as follows:

In operation 1407, the top user application's control module 306 reports completion of the stop children request to the executive 210. The executive 210 then requests the root

J2EE Server 1052 to deallocate the top user application's control module 306, in operation 1408. In response, the root J2EE Server 1052 disables requests to the top user application's control module 306, in operation 1409, and writes any unsaved recoverable state of the top user application's control module 306 to the repository 606 ~~state server 614~~, in operation 1410. Then, in operation 1411, the executive 210 ~~root J2EE Server 1052~~ deletes the top user application's control module 306 from its memory, and reports completion to the executive 210, in operation 1412.

Please amend the paragraph beginning on page 30, line 1, as follows:

In operation 1415, the root J2EE Server 1052 disables requests to the executive 210. The executive 210 ~~root J2EE Server 1052~~ then writes any unsaved recoverable state of the executive 210 to the state server 614, in operation 1416, and then deletes the executive 210 from its memory, in operation 1417. Thereafter, the root J2EE Server 1052 reports completion to the bootstrap program 216, in operation 1418.

Please amend the paragraph beginning on page 1, line 12, as follows:

This application is related to (1) U.S. Patent Application No. 09/812,536 ~~(Attorney Docket No. SUNMP002A)~~, filed March 19, 2001, and "Method and Apparatus for Providing Application Specific Strategies to a Java Platform including Start and Stop Policies," (2) U.S. Patent Application No. 09/812,537 ~~(Attorney Docket No. SUNMP002B)~~, filed March 19, 2001, and "Method and Apparatus for Providing Application Specific Strategies to a Java Platform including Load Balancing Policies," (3) U.S. Patent Application No. 09/833,856 ~~(Attorney Docket No. SUNMP004)~~, filed April 11, 2001, and entitled "Method and Apparatus for Performing Failure Recovery in a Java Platform," (4) U.S. Patent Application No. 09/818,214 ~~(Attorney Docket No. SUNMP005)~~, filed March 26, 2001, and entitled

"Method and Apparatus for Managing Replicated and Migration Capable Session State for A Java Platform," and (5) U.S. Patent Application No. 09/025,249 ~~(Attorney Docket No. SUNMP006)~~, filed April 2, 2001, and entitled "Method and Apparatus for Partitioning of Managed State for a Java based Application." Each of the above related application are incorporated herein be reference.

Please amend the paragraph beginning on page 18, line 1, as follows:

Figure 6 is an architectural diagram showing class structure for a Java system 600, in accordance with an embodiment of the present invention. The Java system 600 includes an application runtime having four main classes: Executive 210, J2EE Server 502, Application 302, and Module 304. Objects from these classes execute over a physical configuration 208 of a cluster 602 comprising several nodes 604 to form a distributed computer system. Typically, the Java system 600 has one Executive object 210, and each node 604 in the cluster 602 runs zero or more J2EE Servers 502, which can be a UNIX process running a Java virtual machine (JVM) with the J2EE runtime classes. However, it should be noted that other structural implementations and arrangements are possible as long as appropriate modifications are made for performing runtime supervision in the Java system 600.

Please amend the paragraph beginning on page 36, line 14, as follows:

Next, in operation 1905, the executive 210 requests the new J2EE Server 502b to take over the modules from the old J2EE Server 502a. To perform this operation, the module move sequence 1600 of Figure 16 is used. After taking over each module from the old J2EE Server 502a, the new J2EE server 502b reports completion to the executive 210, in operation 1906. Operations 1905 and 1906 are repeated for all the modules that are allocated to the new ~~old~~ J2EE Server 502b.

Please amend the paragraph beginning on page 6, line 1, as follows:

Broadly speaking, the embodiments of the present invention address these needs by providing systems and methods for performing online upgrades using a control module, executed as part of an application, that includes application-specific strategies for the application, yet can be coded using the JAVA programming language. In one embodiment, a method for performing an online upgrade to a Java application is disclosed. Initially, an application is executed that includes an original service module and an original control module. The original control module includes application-specific policies ~~polices~~ for the application. To upgrade the application, an upgraded control module is generated. Then, the upgraded control module is used to create an upgraded service module. Both the upgraded control module and the upgraded service module can be generated using class files for the upgraded modules, which are loaded from a system repository. In this manner, the original control module and the original service module are upgraded. This method can also be used to upgrade a child application using the upgraded control module, and the application-specific policies ~~polices~~ can be passed to a control module of the child application.

Please amend the paragraph beginning on page 6, line 17, as follows:

In another embodiment, a Java platform capable of performing online software upgrades is disclosed. The Java platform includes an application having an original service module and an original control module. As above, the original control module includes application-specific policies ~~polices~~ for the application. In addition, the Java platform includes a repository having upgraded class files for the original control module and the original service module. When performing an online software upgrade the original control module is upgraded by generating an upgraded control module using the upgraded class files for the original control module loaded from the repository. Then, the original service module

is upgraded by creating an upgraded service module using the upgraded control module. As with the control module, the upgraded service module is generated using upgraded class files for the original service module loaded from the repository. Further, the upgraded control module is capable of initiating the upgrade of a child application, where the application-specific policies ~~polices~~ are passed to a control module of the child application during the upgrade of a child application.

Please amend the paragraph beginning on page 12, line 12, as follows:

The control module of the embodiments of the present invention allows application developers to provide application-specific strategies to their applications without needing to alter the JAVA platform itself, using the JAVA programming language. Thus, developers can define start and stop policies ~~polices~~, load balancing policies, upgrade policies, and other application-specific strategies and policies, as will be apparent to those skilled in the art. When performing upgrades, the control module provides correlation and upgrade policies ~~polices~~ for upgrading related service modules and child applications. Using the embodiments of the present invention, a system can continue to provide services to clients while a software upgrade is being performed. Advantageously, the embodiments of the present invention allow an online upgrade to occur generally without any detectable impact on the clients using the services provided by the application during the upgrade.

Please amend the paragraph beginning on page 48, line 7, as follows:

An invention is disclosed for performing online upgrades using a control module, executed as part of an application, that includes application-specific strategies for the application, yet can be coded using the JAVA programming language. An application is executed that includes an original service module and an original control module. The

original control module includes application-specific policies ~~polices~~ for the application. To upgrade the application, an upgraded control module is generated. Then, the upgraded control module is used to create an upgraded service module. Both the upgraded control module and the upgraded service module can be generated using class files for the original modules, which are loaded from a system repository. In this manner, the original control module and the original service module are upgraded.